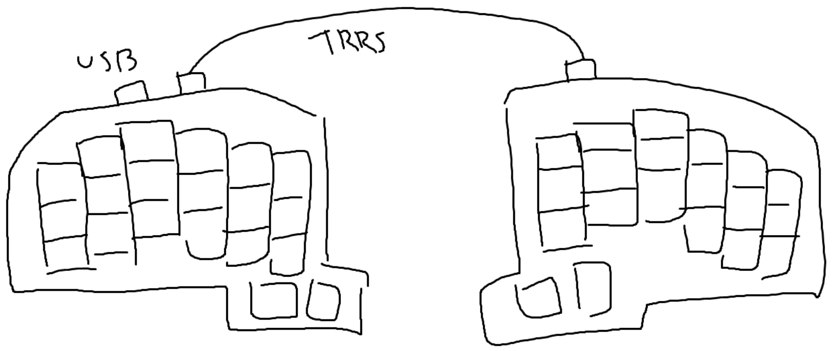
Cloud40 - Designed by Brandon Zhang

Overview

Hi! I'm Brandon, and welcome to my cloud40 build log. This space is dedicated to showcasing the steps I went through to design the cloud40, as well as any thoughts or insights I want to provide to explain the process.

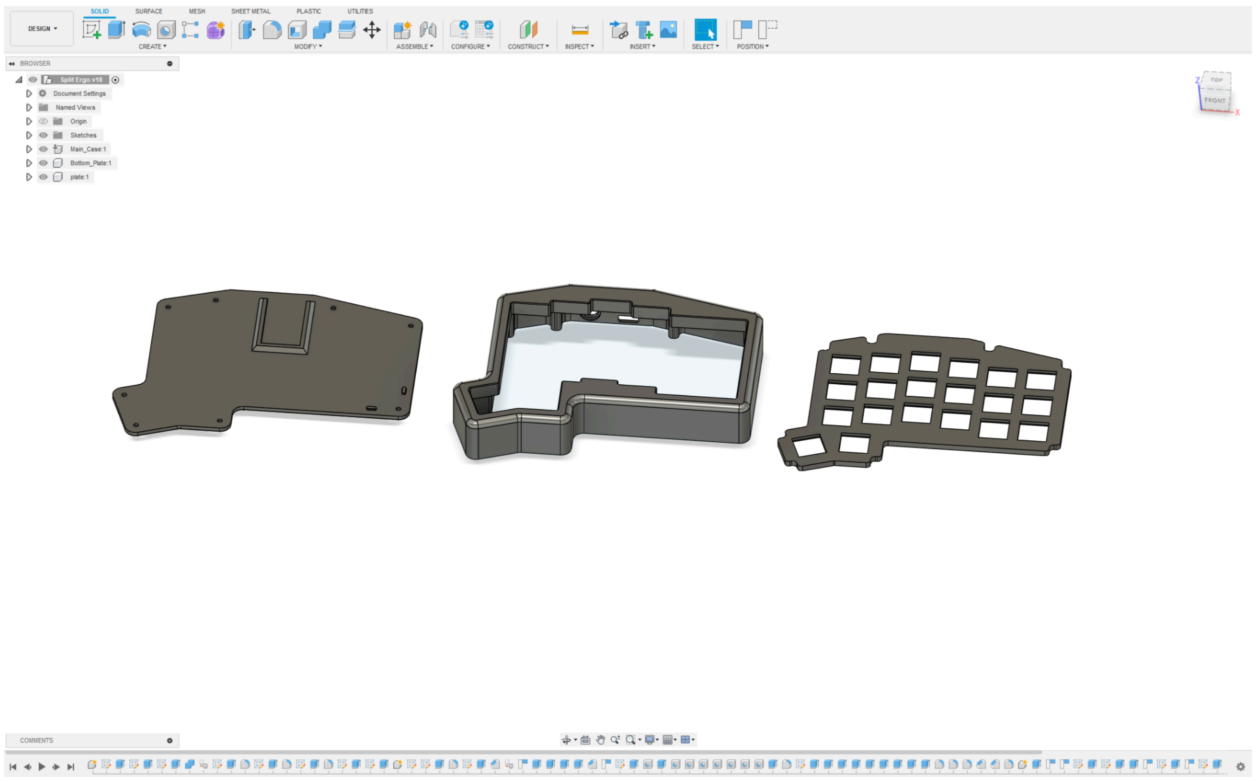
Initial Concept



The cloud40 is a split keyboard that allows the user to position each half wherever they want. The keys are arranged in a grid-like fashion, and the columns are then staggered to match the natural contours of the hand. There are only three rows for each finger to minimize stretching and unnecessary movement. Full functionality is achieved through one of the four thumb keys on the bottom, which toggles between letter, number, and function layers, similar to a phone keyboard. The main goal of this keyboard is to provide a comfortable typing experience for people with arthritis.

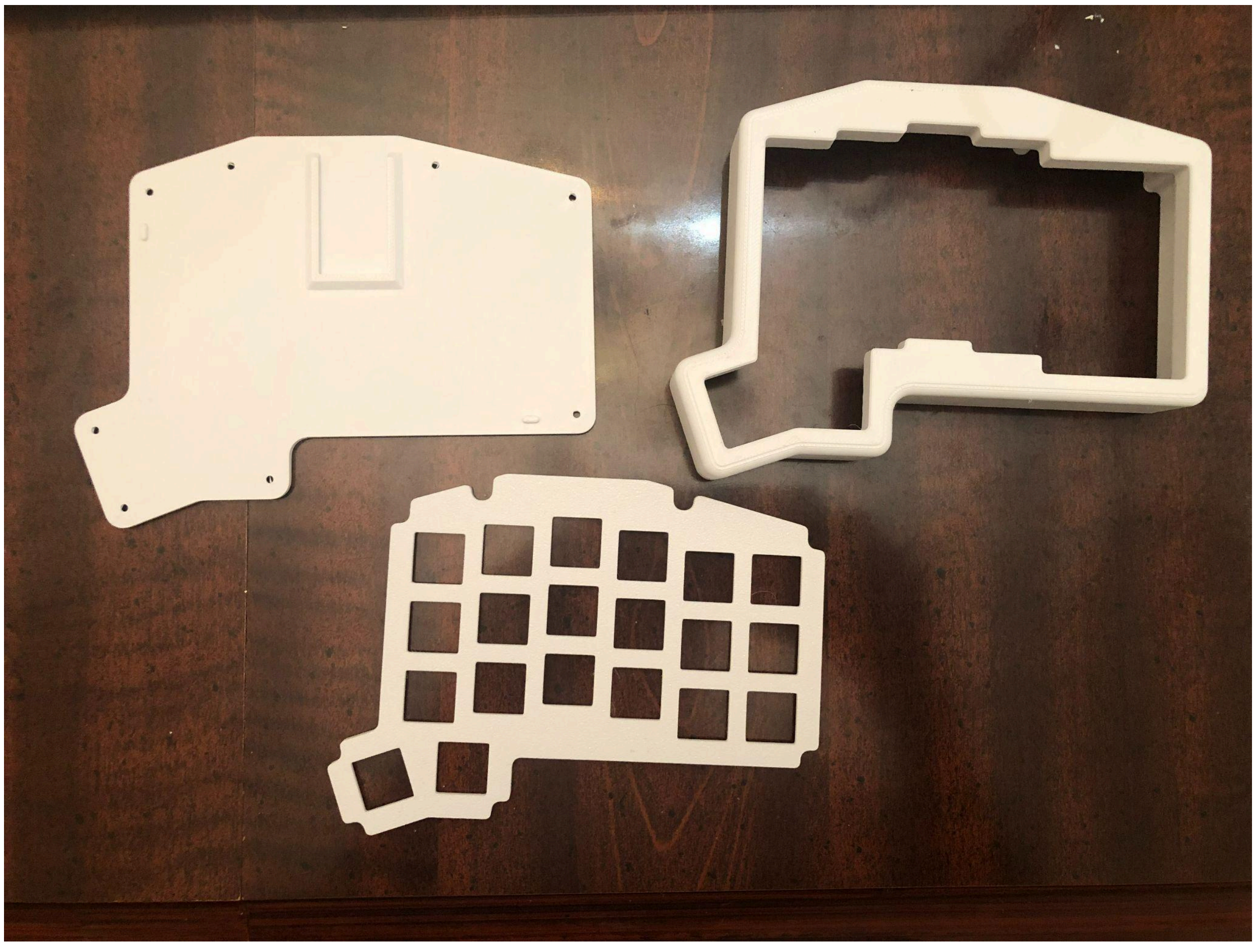
Each half is controlled by an Arduino Pro Micro microcontroller, which detects the keypresses and sends the information to the computer. The two microcontrollers are connected through a TRRS audio cable, since a USB connection poses a risk of the computer mistaking it for a normal connection and shorting out the device.

3D Modeling



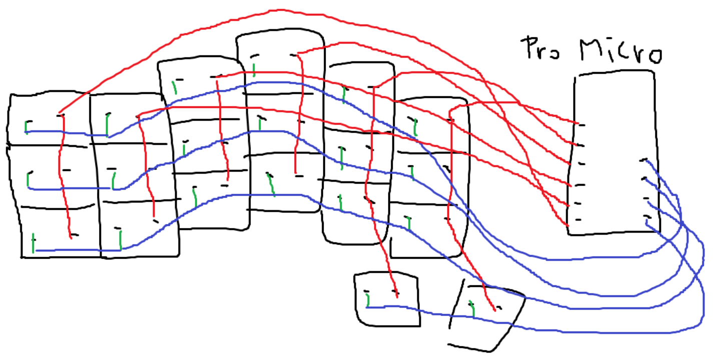
The parts for one half of the keyboard were modeled in Fusion 360. I then mirrored the model in my 3D printer's slicer for the other half.

Prototyping



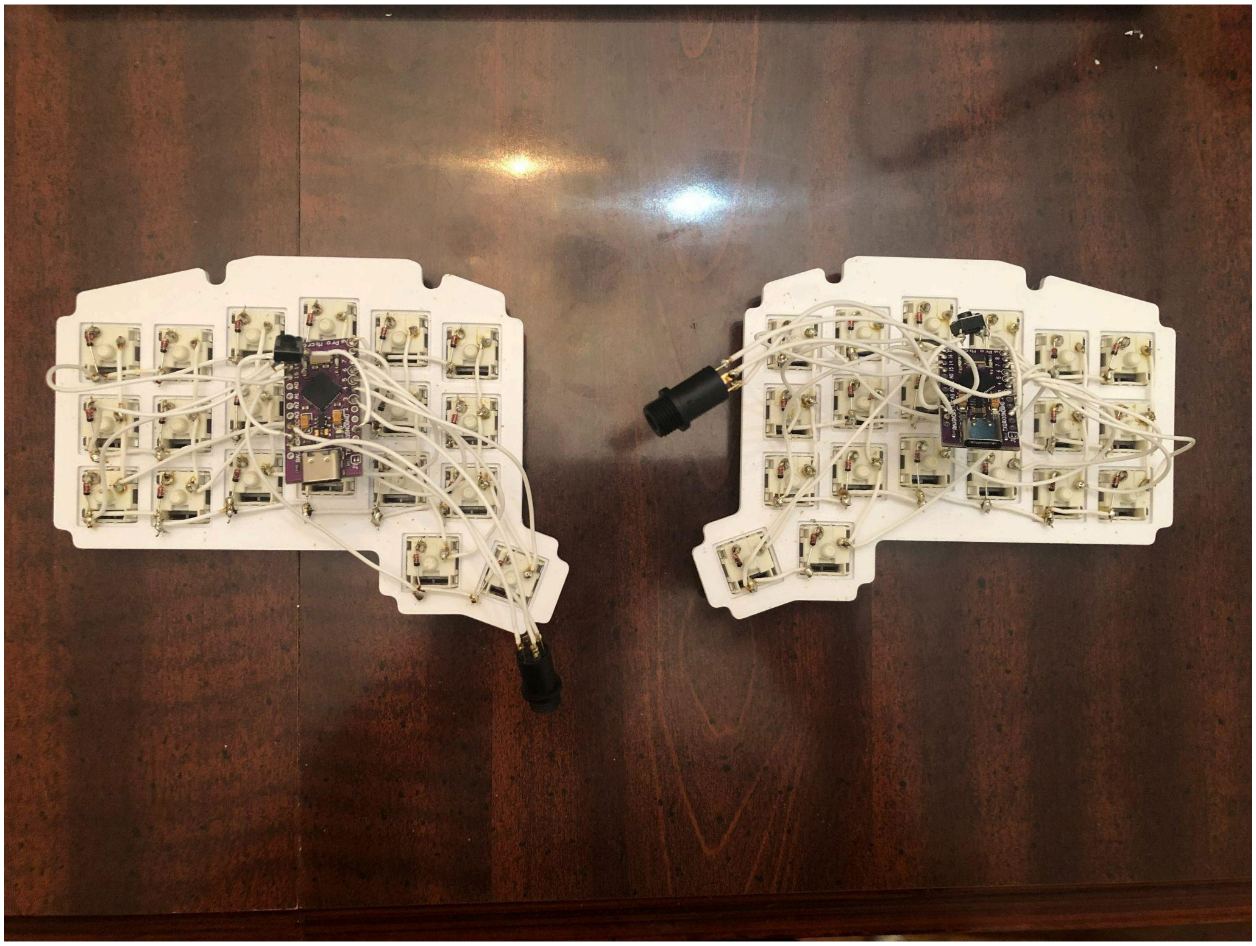
I was already familiar with the dimensions of a mechanical keyboard, so initially test-fitting the switches and keycaps went smoothly. In earlier prints the hanging edges inside the main case were unstable, so I needed to tweak the support settings in the slicer. I also moved the alignment pins on the bottom plate closer together to compensate for the looser tolerance with the top case.

Wiring Matrix

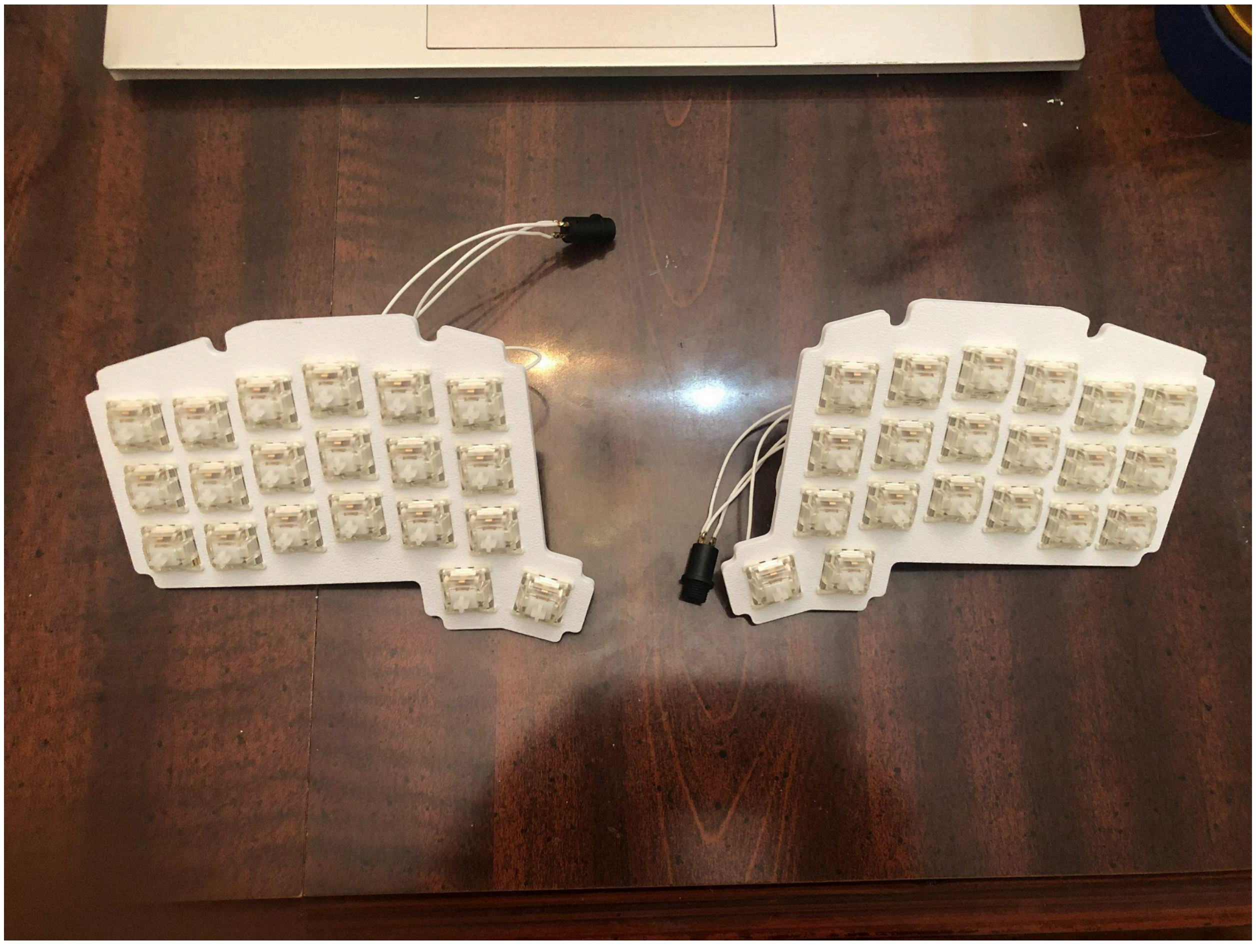


Each key switch has two metal pins on the bottom, which are used to wire them up to a circuit board. However, since this keyboard is wired by hand, each pin needs to have a way to communicate with the central microcontroller. Since the Pro Micro doesn't have enough pins to accommodate all 20 keys individually, each row and each column (shown above in blue and red, respectively) is assigned to a pin. That way, when a key is pressed, the microcontroller knows which row and which column the key is in and can output the correct character. Each row pin has a diode attached to it (shown in green), only allowing the current to flow in one direction. This ensures the microcontroller can correctly detect the input from each individual key if multiple keys are pressed simultaneously.

Assembly



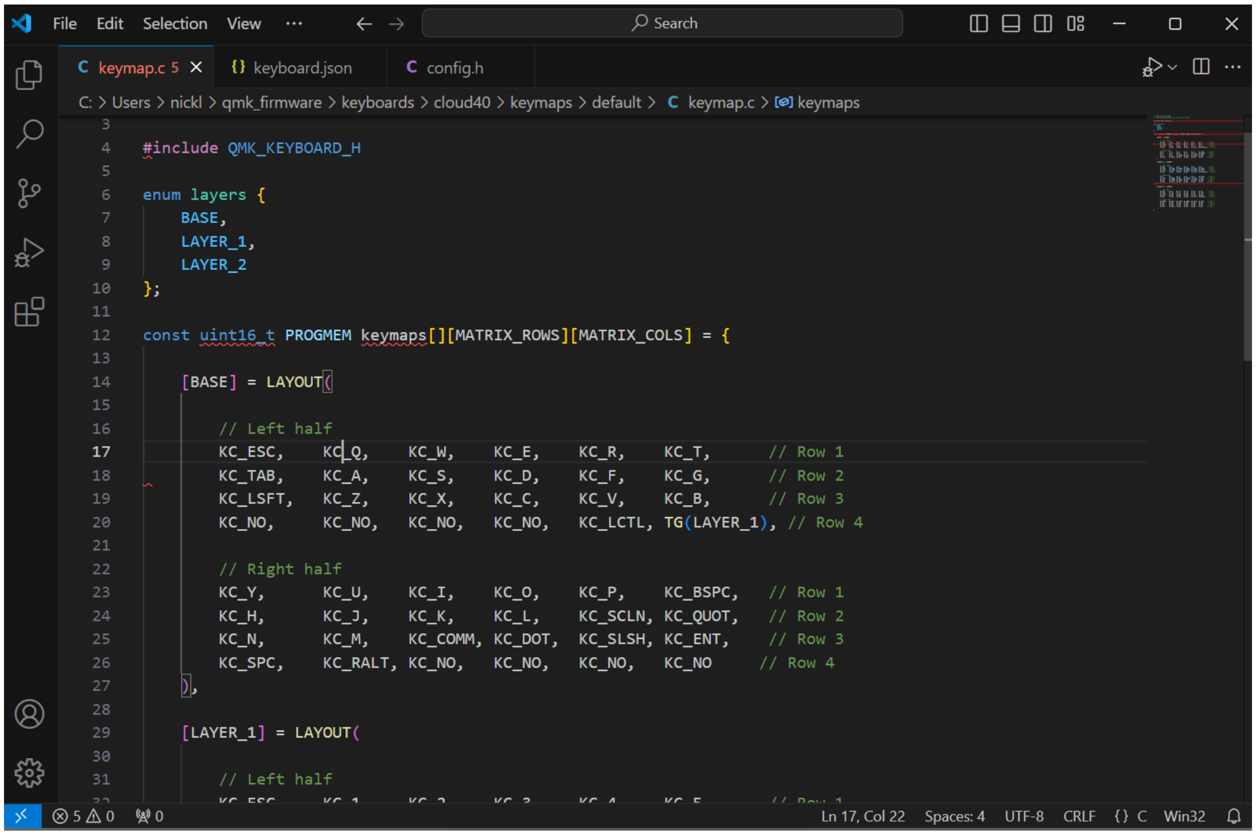
Shown above is the fully soldered assembly. For the sake of convenience, I also wired up two reset buttons to each Pro Micro, so I wouldn't have to use tweezers to touch the pins when I flash the firmware.



The switches I used for this keyboard were the Gateron Clears, which have a pleasantly light operating force of 35g. This should further minimize any pain or discomfort when typing on this keyboard, especially since this design is intended for people with arthritis.

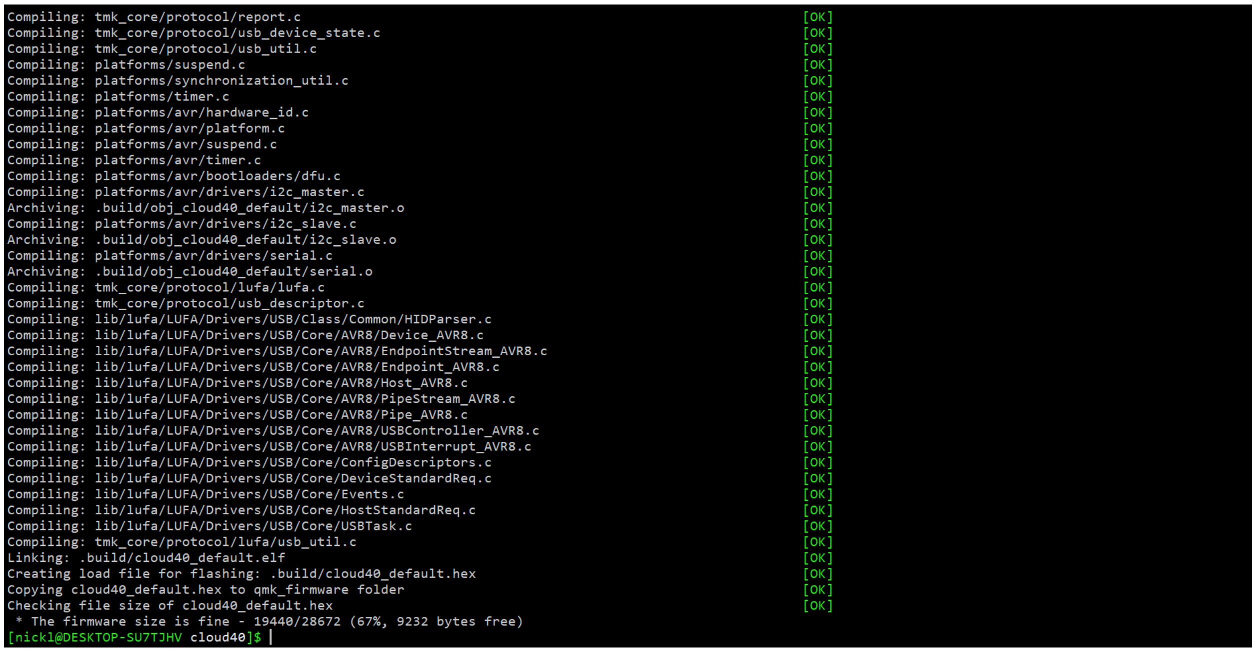
Firmware

The firmware for this keyboard was built using QMK, which converts my code into a .hex file that I can flash onto the Pro Micros (see code below).



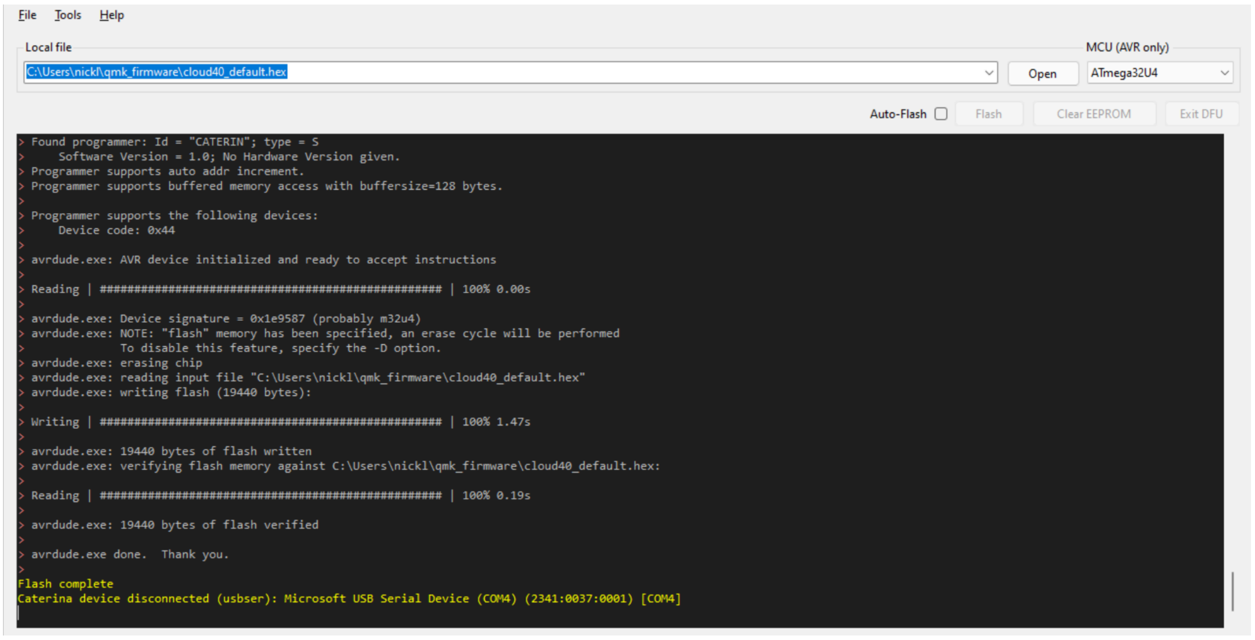
```c

#include QMK_KEYBOARD_H

enum layers {
    BASE,
    LAYER_1,
    LAYER_2
};

const uint16_t PROGMEM keymaps[][MATRIX_ROWS][MATRIX_COLS] = {

    [BASE] = LAYOUT(

        // Left half
        KC_ESC,    KC_Q,    KC_W,    KC_E,    KC_R,    KC_T,        // Row 1
        KC_TAB,    KC_A,    KC_S,    KC_D,    KC_F,    KC_G,        // Row 2
        KC_LSFT,   KC_Z,    KC_X,    KC_C,    KC_V,    KC_B,        // Row 3
        KC_NO,     KC_NO,   KC_NO,   KC_NO,   KC_LCTL, TG(LAYER_1), // Row 4

        // Right half
        KC_Y,      KC_U,    KC_I,    KC_O,    KC_P,    KC_BSPC,     // Row 1
        KC_H,      KC_J,    KC_K,    KC_L,    KC_SCLN, KC_QUOT,     // Row 2
        KC_N,      KC_M,    KC_COMM, KC_DOT,  KC_SLSH, KC_ENT,      // Row 3
        KC_SPC,    KC_RALT, KC_NO,   KC_NO,   KC_NO,   KC_NO       // Row 4
    ),

    [LAYER_1] = LAYOUT(

        // Left half
```

Here is where I create the keymap of the keyboard. Shown above is the base layer, which is the standard qwerty layout along with a few essential modifiers.



```
Compiling: tmk_core/protocol/report.c                                                                   [OK]
Compiling: tmk_core/protocol/usb_device_state.c                                                         [OK]
Compiling: tmk_core/protocol/usb_util.c                                                                 [OK]
Compiling: platforms/suspend.c                                                                          [OK]
Compiling: platforms/synchronization_util.c                                                             [OK]
Compiling: platforms/timer.c                                                                            [OK]
Compiling: platforms/avr/hardware_id.c                                                                  [OK]
Compiling: platforms/avr/platform.c                                                                     [OK]
Compiling: platforms/avr/suspend.c                                                                      [OK]
Compiling: platforms/avr/timer.c                                                                        [OK]
Compiling: platforms/avr/bootloaders/dfu.c                                                              [OK]
Compiling: platforms/avr/drivers/i2c_master.c                                                           [OK]
Archiving: .build/obj_cloud40_default/i2c_master.o                                                      [OK]
Compiling: platforms/avr/drivers/i2c_slave.c                                                            [OK]
Archiving: .build/obj_cloud40_default/i2c_slave.o                                                       [OK]
Compiling: platforms/avr/drivers/serial.c                                                               [OK]
Archiving: .build/obj_cloud40_default/serial.o                                                          [OK]
Compiling: tmk_core/protocol/lufa/lufa.c                                                                [OK]
Compiling: tmk_core/protocol/usb_descriptor.c                                                           [OK]
Compiling: lib/lufa/LUFA/Drivers/USB/Class/Common/HIDParser.c                                           [OK]
Compiling: lib/lufa/LUFA/Drivers/USB/Core/AVR8/Device_AVR8.c                                            [OK]
Compiling: lib/lufa/LUFA/Drivers/USB/Core/AVR8/EndpointStream_AVR8.c                                    [OK]
Compiling: lib/lufa/LUFA/Drivers/USB/Core/AVR8/Endpoint_AVR8.c                                          [OK]
Compiling: lib/lufa/LUFA/Drivers/USB/Core/AVR8/Host_AVR8.c                                              [OK]
Compiling: lib/lufa/LUFA/Drivers/USB/Core/AVR8/PipeStream_AVR8.c                                        [OK]
Compiling: lib/lufa/LUFA/Drivers/USB/Core/AVR8/Pipe_AVR8.c                                              [OK]
Compiling: lib/lufa/LUFA/Drivers/USB/Core/AVR8/USBController_AVR8.c                                     [OK]
Compiling: lib/lufa/LUFA/Drivers/USB/Core/AVR8/USBInterrupt_AVR8.c                                      [OK]
Compiling: lib/lufa/LUFA/Drivers/USB/Core/ConfigDescriptors.c                                           [OK]
Compiling: lib/lufa/LUFA/Drivers/USB/Core/DeviceStandardReq.c                                           [OK]
Compiling: lib/lufa/LUFA/Drivers/USB/Core/Events.c                                                      [OK]
Compiling: lib/lufa/LUFA/Drivers/USB/Core/HostStandardReq.c                                             [OK]
Compiling: lib/lufa/LUFA/Drivers/USB/Core/USBTask.c                                                     [OK]
Compiling: tmk_core/protocol/lufa/usb_util.c                                                            [OK]
Linking: .build/cloud40_default.elf                                                                     [OK]
Creating load file for flashing: .build/cloud40_default.hex                                             [OK]
Copying cloud40_default.hex to qmk_firmware folder                                                      [OK]
Checking file size of cloud40_default.hex                                                               [OK]
 * The firmware size is fine - 19440/28672 (67%, 9232 bytes free)
[nickl@DESKTOP-SU7TJHV cloud40]$ 
```

Compiling the firmware.

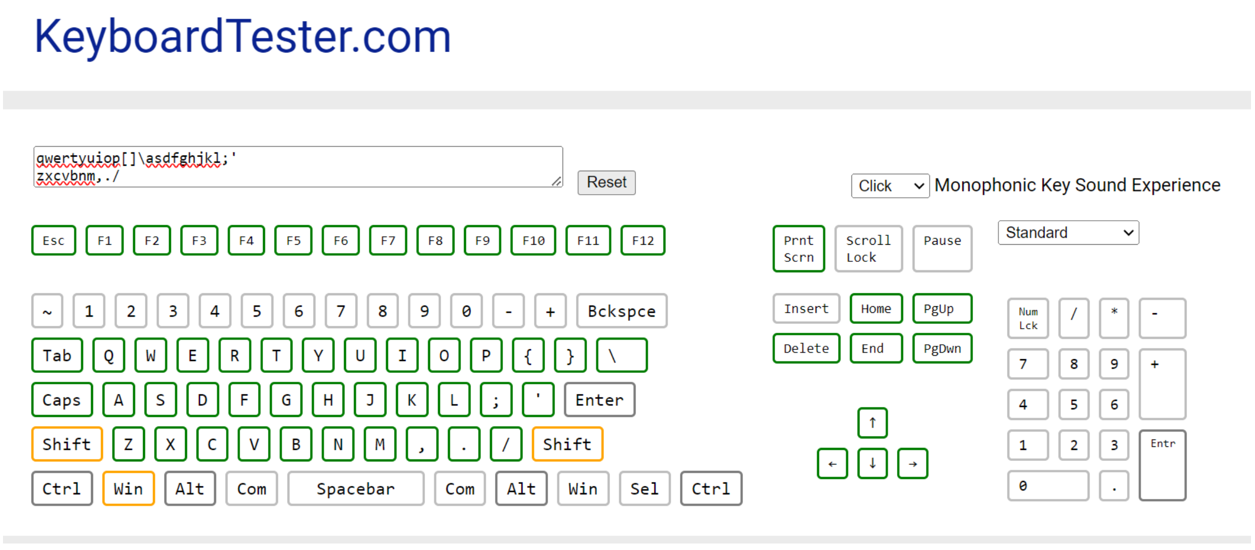


Local file: C:\Users\nickl\qmk_firmware\cloud40_default.hex

MCU (AVR only): ATmega32U4

```
> Found programmer: Id = "CATERIN"; type = S
>     Software Version = 1.0; No Hardware Version given.
> Programmer supports auto addr increment.
> Programmer supports buffered memory access with buffersize=128 bytes.
>
> Programmer supports the following devices:
>     Device code: 0x44
>
> avrdude.exe: AVR device initialized and ready to accept instructions
>
> Reading | ################################################## | 100% 0.00s
>
> avrdude.exe: Device signature = 0x1e9587 (probably m32u4)
> avrdude.exe: NOTE: "flash" memory has been specified, an erase cycle will be performed
>              To disable this feature, specify the -D option.
> avrdude.exe: erasing chip
> avrdude.exe: reading input file "C:\Users\nickl\qmk_firmware\cloud40_default.hex"
> avrdude.exe: writing flash (19440 bytes):
>
> Writing | ################################################## | 100% 1.47s
>
> avrdude.exe: 19440 bytes of flash written
> avrdude.exe: verifying flash memory against C:\Users\nickl\qmk_firmware\cloud40_default.hex:
>
> Reading | ################################################## | 100% 0.19s
>
> avrdude.exe: 19440 bytes of flash verified
>
> avrdude.exe done.  Thank you.
>
Flash complete
Caterina device disconnected (usbser): Microsoft USB Serial Device (COM4) (2341:0037:0001) [COM4]
```

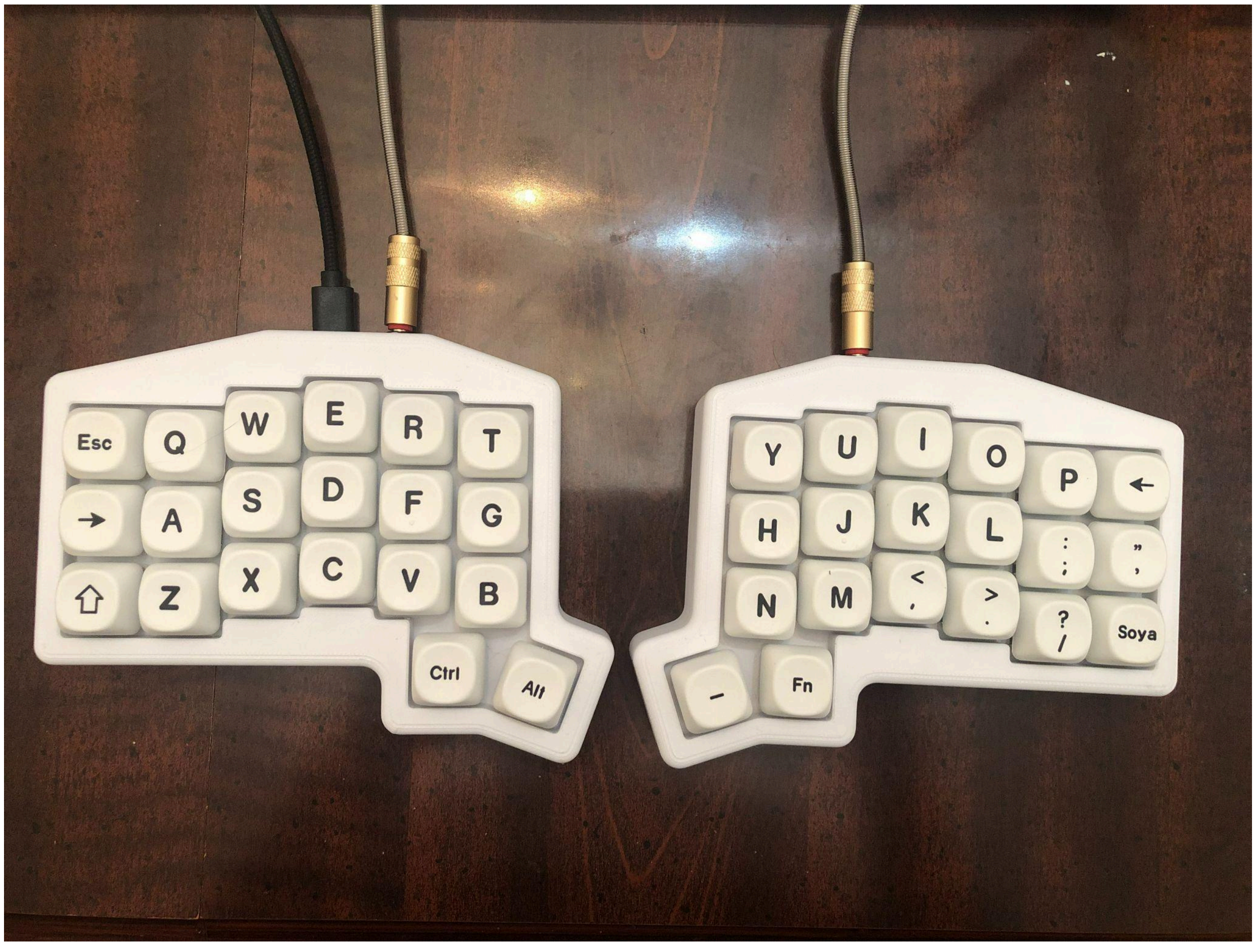
Flashing the firmware using the QMK toolbox app.


KeyboardTester.com
Reset
Click
Monophonic Key Sound Experience
Standard
Esc F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 F11 F12
Prnt Scrn
Scroll Lock
Pause
~ 1 2 3 4 5 6 7 8 9 0 - + Bckspce
Tab Q W E R T Y U I O P { } \
Caps A S D F G H J K L ; ' Enter
Shift Z X C V B N M , . / Shift
Ctrl Win Alt Com Spacebar Com Alt Win Sel Ctrl
Insert Home PgUp
Delete End PgDwn
↑
← ↓ →
Num Lck / * -
7 8 9 +
4 5 6
1 2 3 Entr
0 .

```
qwertyuiop[]\asdfghjkl;'
zxcvbnm,./
```

Success!

The Final Keyboard



Here’s the finished keyboard. I chose a set of keycaps that have a uniform height, allowing for easy interchange if the user wants to switch keymaps.